Compass Sensor for Lego RCX

never get lost again

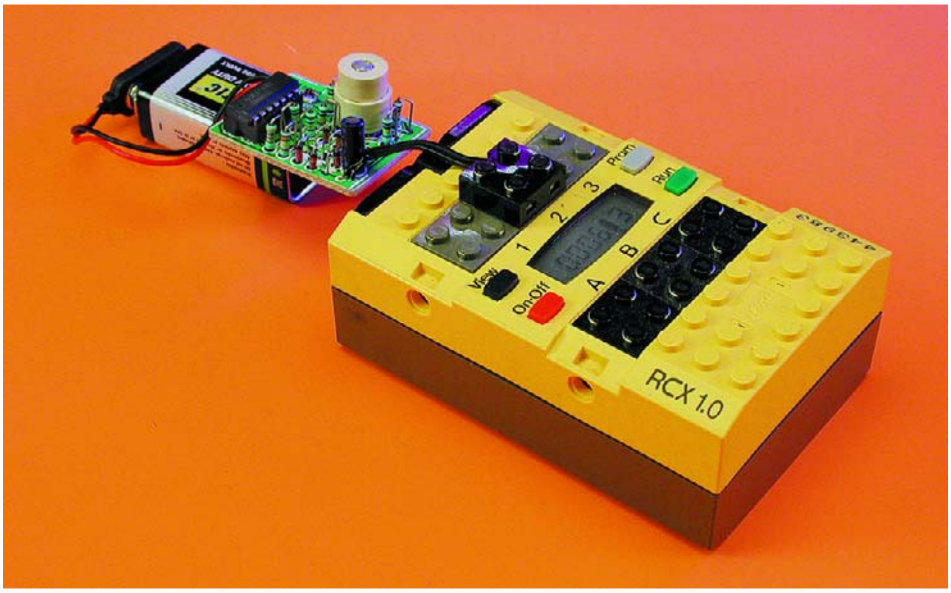
Design by Z. Ottten, write-up by L. Lemmens

Mindstorms, the by now well-known series of robot construction materials from Lego, has already been the subject of several articles in *Elektor Electronics*. In 2000, we presented a series of five articles dealing with all the ins and outs of the RCX module, which is a microcontroller block with three sensor inputs and three outputs for controlling items such as motors and lamps. Following this came a light sensor, a proximity sensor and a sensor multiplexer, and recently a design for an I^{2}C interface appeared in our magazine. This month it's time for a compass sensor, which enables us to give our robots a sense of direction.

If a robot has to travel through a space, this can be implemented in a variety of manners. The first option is to allow the robot to search out its own path. Proximity and contact sensors allow the robot to independently avoid obstacles in order to prevent damage and/or prevent the robot from ending up in a situation that it cannot get out of on its own. The Mindstorms package includes the sensors that we need for this approach.

A second option is to give the robot an objective: to allow it to independently find a way to reach a previously specified location. One example is following a line that defines the course to be taken, such as the one on the mat included in this Lego box.

The route can also be defined in the RCX module by energising the robot's motors for defined intervals and calculating the path that has been travelled during these intervals. The disadvantage of this method is that the speed of the motors depends on the voltage of the RCX battery. In the course of time, the motors will turn more slowly. This causes a gradual decrease in the accuracy of position determination. A compass is an excellent means to at least measure the direction in which the robot is moving, in order to allow the robot to determine its orientation and modify it as necessary. In this article, we describe a



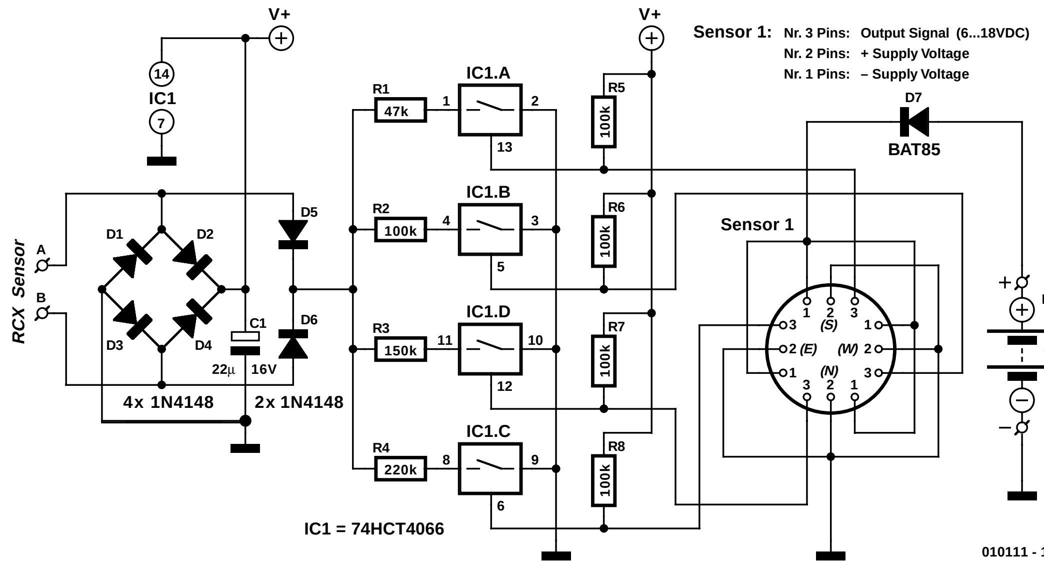


Figure 1. The circuit diagram consists of four analogue switches, a few passive components and a Pewatron type 6945 compass sensor.

simple manner to equip a Mindstorms robot with a compass.

Our design uses a Pewatron type 6945 digital compass sensor. This is not the first time that this sensor has been used in an *Elektor Electronics* project, since in September 1996 we presented an electronic compass that also uses the 6945. The sensor contains a miniature rotor, a magnet and a special Hall-effect IC, which are used together to determine its orientation with respect to the earth's magnetic field. The four digital outputs, of which two can be simultaneously active, provide a resolution of 45 degrees for determining the sensor orientation. It should be noted that the sensor must be aligned perpendicular to the surface of the earth and that external magnetic fields can significantly disturb the measurement. At a price of around €50 (approx. £30), the 6945 is not exactly cheap, so it should be handled as carefully as possible.

The circuit

The circuit diagram of our RCX compass is shown in **Figure 1**. The circuit that provides the interface between the sensor and the RCX module is powered from the RCX module. As you probably know, the RCX can power a sensor and make measurements on the same sensor using only two leads (see **Figure 1**). Most of the time, the supply voltage is present on the A and B terminals. During short intervals, the supply voltage is switched off and a measurement is made. During the measurement interval, the interruption in the supply voltage is bridged by electrolytic capacitor C1. Diode bridge D1–D4 ensures that terminals A and B can be swapped without causing any problems. During the measurement interval, diodes D5 and D6 play a comparable role in determining the resistance of the circuit connected to the RCX input.

The actual sensor, the 6945, draws a fairly large current (20 mA) and is thus powered by the 9-V battery Bt1, since the sensor terminals of the RCX module can supply a maximum current of only 10 mA and would otherwise be overloaded. Diode D7 protects the sensor against a reversed-polarity supply voltage.

The 6945 has four open-collector outputs, of which two are concurrently active (Low), depending on the orientation of the sensor. The pull-up resistors R5–R8 adapt the voltage levels at the sensor outputs to the supply voltage of the interface circuit, which as previously mentioned is powered by the RCX module.

The four outputs of the 6945 drive the four analogue switches of IC1, which in turn cause a certain combination of resistors R1–R4 to be connected in parallel to the RCX sensor input. **Table 1** shows the values measured by the RCX module, depending on the orientation of the sensor. Here we see eight compass directions as a function of the state of the compass sensor. Note that only a small portion of the measurement range of the RCX is used, but this is adequate for determining the eight states of the sensor with sufficient accuracy (the RCX can determine a measurement value between 0 and 1023 at its input).

It would have probably been possible to connect resistors R1–R4 directly to the outputs of the sensor, which would allow IC1 and R5–R8 to be eliminated. However, considering the price of the sensor and the fact that the sensor data sheet is very skimpy and gives hardly any information about the load capacity of the outputs, we decided we would rather be safe than sorry.

Construction

If you want to have a truly compact sensor, you best bet is to build the circuit on a piece

Table 1.

Recommended values for resistors R1–R4

Direction	sensor pin 3	6	9	12	RCX value
N	1	1	1	0	863
NO	0	1	1	0	920
O	0	1	1	1	810
ZO	0	0	1	1	856
Z	1	0	1	1	818
ZW	1	0	0	1	830
W	1	1	0	1	787
NW	1	1	0	0	887

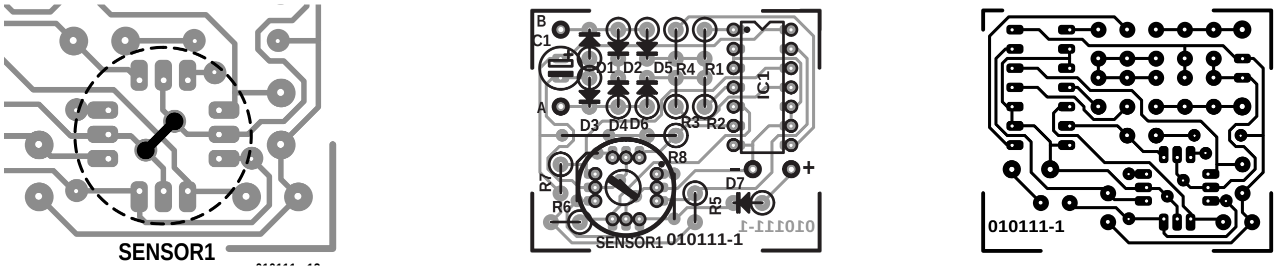


Figure 2. In order to keep the circuit board small, the resistors and diodes are fitted vertically. Note the wire bridges next to the sensor! (board not available ready-made).

of perforated prototyping board in order to keep the dimensions of the circuit as small as possible. However, if you would like a bit more convenience and are not all that worried about size, you can use the circuit board layout shown in **Figure 2**. Pay attention to the wire bridges next to the sensor, especially the short wire bridge under the middle of the sensor. The best approach is to use insulated wire for these bridges and fit them to the copper side of the board, since this allows the sensor to be fitted flush to the surface of the board.

For connecting the sensor to the RCX module, we use a standard Lego cable with terminal bricks, which we cut in two. The battery should be mounted as far away from the sensor as possible, since otherwise the metal case of the battery will affect the reading.

Listing 1. Sample program in Visual Basic.

```
Private Sub Command1_Click()
Const setpoint = 5
Const measure = 6
Const dif = 7

With RCX
.ComPortNo = 1  ‘ COM port 1
.LinkType = 0   ‘ 0 = IR
.PBrick = 1  ‘ 1 = RCX, 0 = CyberMaster
.InitComm          ‘ Initialise the COM port

.SelectPrgm SLOT3
.BeginOfTask MAIN
    .SetSensorType SENSOR_2, LIGHT_TYPE
    .SetSensorMode SENSOR_2, RAW_MODE, 0

    .SetVar setpoint, CON, 1 ‘1=N,2=NE,3=E,4=SE,5=S,6=SW,7=W,8=NW
    .SetVar measure, CON, 0
    .SetVar dif, CON, 0

    .StartTask 1
.EndOfTask ‘Main

.BeginOfTask 1
.Loop CON, forever
‘North
    .If SENVAL, SENSOR_2, EQ, CON, 863
    .SetVar measure, CON, 1
    .EndIf
    .If SENVAL, SENSOR_2, EQ, CON, 862
    .SetVar measure, CON, 1
    .EndIf
    .If SENVAL, SENSOR_2, EQ, CON, 861
    .SetVar measure, CON, 1
    .EndIf
‘North-East
    .If SENVAL, SENSOR_2, EQ, CON, 920
    .SetVar measure, CON, 2
    .EndIf
    .If SENVAL, SENSOR_2, EQ, CON, 919
    .SetVar measure, CON, 2
    .EndIf
    .If SENVAL, SENSOR_2, EQ, CON, 918
    .SetVar measure, CON, 2
    .EndIf
```

Software

The sample program (see **Listing 1**) for using our compass sensor is written in Visual Basic. When the software provided on the Mindstorms CD is installed, the SPIRIT.OCX software library is also installed. This library allows the RCX to be programmed and controlled using this high-level language. The listing can also be used with other languages, such as Delphi or C++, without major modifications.

Starting with a blank form in Visual Basic, we place a Spirit control and a Command button on it. A double click on this button opens a window into which we can copy Listing 1. When we execute the Visual Basic program, the application is sent to the RCX block via the Tower.

The sample program is reasonably simple. In the program, a direction is specified in the line *SETVAR setpoint, CON, 1*. The latter number can be assigned a value from 1 to 8 in order to set the desired direction. Following this, the value of the sensor is read using an infinite loop. If the orientation matches the previously set direction, a tone is emitted

COMPONENTS LIST

Resistors:
R1 = 47kΩ
R2,R5-R8 = 100kΩ
R3 = 150kΩ
R4 = 220kΩ

Capacitors:
C1 = 22µF 16V radial

Semiconductors:
D1-D6 = 1N4148
D7 = BAT85
IC1 = 74HCT4066

Miscellaneous:
Sensor1 = Pewatron 6945 (Pewatron AG, www.pewatron.com)
BT1 = 9V PP3 battery with clip-on lead
Lego connection cable (cut in two):
length 26.6cm, Lego order code 5311
length 9cm, Lego order code 5041

by the RCX buzzer.

The four lines following the statement *With RCX* are related to the communications between the PC and the RCX module. They have no direct significance for the actual application. Following this, program slot 3 in the RCX module is selected. After this, sensor 2 is configured in the task *Main* so we can work with the compass sensor, the desired direction is selected (*setpoint*) and the registers for storing the measured value (*measure*) and the difference between the measured and target values (*dif*) are set to zero. The final statement in *Main* starts task 1, which contains the infinite loop.

When a measurement is made, three values are checked for each direction in order to compensate for any variations that may be present. If the measured value actually falls within the range of a particular compass direction, the *measure* register receives a value equal to the sequence number of this direction.

After these 24 comparisons with the measured value have been made, a check is made to see whether the measured direction matches the set direction. If this is the case (*dif=0*), routine *Playsystemsound* is executed.

(010111-1)

```
‘East
    .If SENVAL, SENSOR_2, EQ, CON, 810
    .SetVar measure, CON, 3
    .EndIf
    .If SENVAL, SENSOR_2, EQ, CON, 809
    .SetVar measure, CON, 3
    .EndIf
    .If SENVAL, SENSOR_2, EQ, CON, 808
    .SetVar measure, CON, 3
    .EndIf
‘South-East
    .If SENVAL, SENSOR_2, EQ, CON, 856
    .SetVar measure, CON, 4
    .EndIf
    .If SENVAL, SENSOR_2, EQ, CON, 855
    .SetVar measure, CON, 4
    .EndIf
    .If SENVAL, SENSOR_2, EQ, CON, 854
    .SetVar measure, CON, 4
    .EndIf
‘South
    .If SENVAL, SENSOR_2, EQ, CON, 818
    .SetVar measure, CON, 5
    .EndIf
    .If SENVAL, SENSOR_2, EQ, CON, 817
    .SetVar measure, CON, 5
    .EndIf
    .If SENVAL, SENSOR_2, EQ, CON, 816
    .SetVar measure, CON, 5
    .EndIf
‘South-West
    .If SENVAL, SENSOR_2, EQ, CON, 830
    .SetVar measure, CON, 6
    .EndIf
    .If SENVAL, SENSOR_2, EQ, CON, 829
    .SetVar measure, CON, 6
    .EndIf
    .If SENVAL, SENSOR_2, EQ, CON, 828
    .SetVar measure, CON, 6
    .EndIf
‘West
    .If SENVAL, SENSOR_2, EQ, CON, 787
    .SetVar measure, CON, 7
    .EndIf
    .If SENVAL, SENSOR_2, EQ, CON, 786
    .SetVar measure, CON, 7
    .EndIf
    .If SENVAL, SENSOR_2, EQ, CON, 785
    .SetVar measure, CON, 7
    .EndIf
‘North-West
    .If SENVAL, SENSOR_2, EQ, CON, 887
    .SetVar measure, CON, 8
    .EndIf
    .If SENVAL, SENSOR_2, EQ, CON, 886
    .SetVar measure, CON, 8
    .EndIf
    .If SENVAL, SENSOR_2, EQ, CON, 885
    .SetVar measure, CON, 8
    .EndIf

‘The sensor value is stored in register ‘measure’
‘the target value in ‘setpoint’
.SetVar dif, VAR, measure ‘dif=measure-setpoint
.SubVar dif, VAR, setpoint
.If VAR, dif, EQ, CON, 0
    .PlaySystemSound 0
.EndIf
.EndLoop
.EndOfTask
End With

End Sub
```